Betulla Rescue
Interactive Graphics Project

Leonardo Brizi
Tommaso Carlini
Eleonora Chiarantano

September 22, 2019

Introduction

Betulla Rescue is the result of the collaboration of three students from La Sapienza University, with the aim of creating a game that was intuitive, fun and dealt with real issues. Within the game the player is the pilot of a Bombardier-415, better known as Canadair, an aircraft supplied to firefighters to fight large fires in areas difficult to reach. The goal is to limit and then extinguish a fire in the shortest possible time. The player will be supported during the game by the co-pilot Camil, who will provide valuable information throughout the mission, will strive to give advice on altitude, speed, commands to press and much more. The game ends when the fire is completely extinguished.

Environment

Libraries

The world and the plane were created using the library three.js[1], a library of Javascript used to create and display animated 3D computer graphics in a web browser. Within three.js we have subsequently imported the following libraries:

- *dat.gui.min.js*: A lightweight graphical user interface for changing variables in JavaScript.
- *GLTFLoader.js*:glTF (GL Transmission Format) is an open format specification for efficient delivery and loading of 3D content.
- *MTLLoader.js*: The Material Template Library format (MTL) or .MTL File Format is a companion file format to .OBJ that describes surface shading (material) properties of objects within one or more .OBJ files.
- *OBJLoader.js*: The OBJ file format is a simple data-format that represents 3D geometry in a human readable format.
- *TextureLoader.js*: Class for loading a texture.

Models

Bombardier-415

The 3D model of the Bombardier-415[2] was downloaded through the site *Cgtrader*[3] and then modified using the program Blender v-2.80[4]. The model is composed of 78 children, which compose the torso, the wings, the propellers, the tail and the landing gear. We chose this model because it faithfully reproduces the object we wanted to recreate and it is easily modeled

Listing 1: Bombardier-415 loader

```
GLTFloader.load( 'Models/Bombardier-415/bombardier_canadair.glb', function ( gltf ) {
          model = gltf.scene;
```

[1]https://threejs.org
[2]All the informations about the Bombardier-415 are taken from http://www2.malignani.ud.it/WebEnis/aer/Portfolio/CANADAIR_CL415.pdf
[3]https://www.cgtrader.com/free-3d-models/aircraft/historic/bombardier-415-superscooper
[4]https://www.blender.org/

Figure 1: Bombardier-415 opened with Blender v-2.80

```
model.traverse(function (children){
    if ( children instanceof THREE.Mesh )
        { children.castShadow = true; }
});
if(!light_mode){
    canadair2 = model.clone();
    canadair3 = model.clone();
    canadair2.position.set(-300, 5, -60);
    canadair2.scale.set(1, 1, 1);
    canadair2.rotation.y = Math.PI * 0.5;
    canadair2.castShadow = true;
    scene.add(canadair2);
    canadair3.position.set(-5, 5, 50);
    canadair3.scale.set(1, 1, 1);
    canadair3.rotation.y = -Math.PI *0.5;
    canadair3.castShadow = true;
    scene.add(canadair3);
    aeroporto.push(canadair2);
    aeroporto.push(canadair3);
}
model.position.set(10, 4, 9);
model.scale.set(1, 1, 1);

// Rename the children
model.traverse(function (children){
    if (children.name == "heliceG") elica_sx = children;
    if (children.name == "heliceD") elica_dx = children;
```

```
            if (children.name == "voletG") flap_int_sx = children;
            if (children.name == "voletD") flap_int_dx = children;

            if (children.name == "aileronG") flap_ext_sx = children;
            if (children.name == "aileronD") flap_ext_dx = children;

            if (children.name == "profondeur") flap_timone = children;
            if (children.name == "direction") timone = children;

            if (children.name == "bolG") bulbo_sx = children;
            if (children.name == "bolD") bulbo_dx = children;

            if (children.name == "porteG") carrello_ant_sx = children;
            if (children.name == "porteD") carrello_ant_dx = children;

            if (children.name == "trappeG") carrello_pst_sx = children;
            if (children.name == "trappeD") carrello_pst_dx = children;

            if (children.name == "roueA") ruote_ant = children;
            if (children.name == "roueG") ruote_pst_sx = children;
            if (children.name == "roueD") ruote_pst_dx = children;

            if (children.name == "axeA") asse_ant = children;
            if (children.name == "axeG1") sospensione_1_sx = children;
            if (children.name == "axeG2") sospensione_2_sx = children;
            if (children.name == "axeG3") sospensione_3_sx = children;
            if (children.name == "axeG4") sospensione_4_sx = children;
            if (children.name == "axeD1") sospensione_1_dx = children;
            if (children.name == "axeD2") sospensione_2_dx = children;
            if (children.name == "axeD3") sospensione_3_dx = children;
            if (children.name == "axeD4") sospensione_4_dx = children;

            if (children.name == "parapG1") supp_carrello_basso_sx = children;
            if (children.name == "parapG2") supp_carrello_alto_sx = children;
            if (children.name == "parapD1") supp_carrello_basso_dx = children;
            if (children.name == "parapD2") supp_carrello_alto_dx = children;
        });
        model.add( camera );
        model.add( particleSys );
        scene.add( model );
    },
    // called while loading is progressing
    function ( xhr ) {
        console.log( ( xhr.loaded / xhr.total * 100 ) + '% loaded' );
    },
    // called when loading has errors
    function ( error ) {
        console.log( 'An error happened' );
    });
```

World

The world generates itself autonomously with the advance of the plane so as to make the game lighter and allow to have a more fluid graphics. We decided to generate a forest not too dense, considering the creation of a multitude of trees too heavy and unnecessary for the final purpose of the game. Our goal was to make the idea of an open wooded area. The position of the lake is fixed and is located in front of the runway so as to make immediate his vision. The fire is generated autonomously in a random position within the "work area". The latter is a limited portion of the world within which the plane can move freely, but that, if the player violates the boundaries, produces the failure of the player and consequently the defeat. In this way we were able to create a world limited in space so as to have a light code and prevent the player from getting lost in the infinite immensity of a virtual game.

Trees We imported the trees from this site. We also decided to manage two different types of trees, a first type is formed by pines[5], and is the main part of our forest. The other type is generated in the areas affected by the fire and is composed of burnt trees[6]. This allows us to make the world more realistic and gives an intuitive idea of the area affected by the fire even when the fire has been extinguished.
The trees are loaded in the world dinamically because if we load the entire forest at the beggining of the game the performace will decreace dramatically. The dinamic loading of the forest consist on load only the threes which are in a range around the airplane.

Grass For the representation of the grass we decided to reproduce a more detailed type of grass[7] only when the plane is near the ground (i.e. during the take-off phase). Otherwise, the ground is covered with a "carpet" that gives the idea of a turf in 2D and therefore much lighter.
It is positioned only near the airport and also for the grass, as well as trees, the loading in the scene depends on the position of the airplane

Fire The fire is composed by four orthogonal plane where on each of them is applyed the fire texture. The texture is procedural and it is generated by the shader. We import the shader from the ThreeJS examples[8].
The fire is controlled by three parameters which depend on the difficulty chosen. The parameters are: FireSpeed, FireInitialDimension, FireInterval. The FireInitialDimension is the initial scale of the fire when the game starts, the FireSpeed is the amount of how much the fire increases each time it expands and the FireInterval is how often it expands. Each time it expands its scale grow up by a factor thath depends on the FireSpeed.
When the water is dropped over the fire, it will reduce its scale. When it reaches a scale below 0.2 the game is over and you have won.

Water The lake consists of a large circle on which is applyed the water texture. Its appearance changes according to the mode chosen by the player: while in light mode it is simply a dark blue expanse, in normal mode it is rapresented in a more realistic way, as its surface reflects the surrounding environment and lights. In order to implement the latter, we took inspiration from one three.js example[9].

[5]https://sketchfab.com/3d-models/pine-tree-single-01-ed72511b36c14446a1b596b7e3686d73
[6]https://sketchfab.com/3d-models/tree-free-extra-details-scan-cdb8c99aa5774fcab712f9082b061a9d
[7]https://sketchfab.com/3d-models/yet-another-grass-model-95dc9c01a9ed45a99a3303b95c9f3f91
[8]https://threejs.org/examples/webgl_fire.html
[9]https://threejs.org/examples/webgl_shaders_ocean.html

After filling the tank on the lake, the player can empty it at any time, as long as the plane is straight-oriented. The water collected is then released through a multitude of drops, each one with its 3D position and speed. All together they form a waterfall, which continues until the tank bar is completely emptied. In order to minimize the computation needed for this animation, we have limited the number of implemented drops by reinitializing at the base of the waterfall the dropped particles that are no longer visible (out of screen). This part is been implemented by taking in consideration some online examples[10].

Airport

Hangars From the websites *Sketchfab* and *Cgtrader*, we decided to import two different types of hangars so as to give a greater variety to the environment and make the airport area more realistic. The first type of hangar is a closed building[11], very similar to a protected office or warehouse. The second is the classic hangar[12] inside the airports, a metal structure that serves to cover the aircraft from the rain and sun. A control tower[13] has been positioned next to the hangars for communication with the base. Hangars, as well as trees, have a dynamic loading. They are loaded in the scene only when they enter in the field of view of the airplane.

Take-off Runway The runway was imported by *Sketchfab*[14] and was designed to allow the aircraft to reach a speed of about 215 km/h. Considering that the stall speed of a Bombardier-415 is about 150 km/h and that therefore it must be the minimum speed to face a takeoff, the runway has been designed long enough to allow the possibility of a safe takeoff. We have also chosen to position the runway in the direction of the lake so as to help the player to reach the water surface as quickly as possible.

Audio

We imported several audio tracks to reproduce the most common sounds. The noise of the motors[15] is activated when the M button is pressed and increases or decreases the frequency as the speed increases or decreases. This gives a good idea of acceleration or deceleration. To close the landing gear we relied on an audio track[16], slowing down the playback to synchronize the sound with the our landing gear closing time. For the fire, we relied on a specific track[17] to reproduce its noise, and we decided to develop it as a positional audio in order to help the player in localizing the fire just by ear. The sound of a waterfall[18] is reproduced each time the plane empties the tank. We decided to also insert a background music track[19] during the initial and pause screens and two mini-tracks to accompany the victory[20] or game over screen[21].

[10] https://github.com/free-unife/threejs-waterfall
[11] https://sketchfab.com/3d-models/hangar-558d7e381b9a4b099a49e72ff5824ecb
[12] https://www.cgtrader.com/free-3d-models/military/other/aircraft-low-poly-shelter-hangar
[13] https://open3dmodel.com/download/communication-tower-building_12469.html
[14] https://sketchfab.com/3d-models/sidewalk-c4d846ab2beb40e09582ef5d93746e6a
[15] https://freesound.org/people/craigsmith/sounds/438650/
[16] https://freesound.org/people/77Pacer/sounds/425273/
[17] http://soundbible.com/1902-Fire-Burning.html
[18] https://freesound.org/people/reinsamba/sounds/24054/
[19] https://www.bensound.com/royalty-free-music/track/badass
[20] http://www.orangefreesounds.com/winning-sound-effect/
[21] https://freesound.org/people/themusicalnomad/sounds/253886/

Animations of the plane

Figure 2: A screenshot during the game

Maneuverability

Although we had initially decided to make the piloting of the aircraft very similar to a real simulation, with time and several tests we realized that the piloting was extremely complicated for a player without aeronautical knowledge. This problem of piloting would be boundless in the absence of fun, the primary objective of the game, and as a result we decided to structure the piloting in a simpler and more intuitive way. We rely on the WASD command set for piloting, i.e. controlling roll[22] and pitch[23] angles. We decided to avoid rotations along the yaw axis because we integrated the movements along this axis in the (automatic) movements of the rudder, which moves according to the turn to counteract the phenomenon of adverse yaw[24]. To turn, the player simply press the A (left) or D (right) commands. Pressing one of these two commands will cause the player to rotate along the roll axis in the direction of the turn, lowering the internal flaps of the internal wing and slightly raising the internal flaps of the external wing to the rotation. This movement of the flaps allows the inclination of the plane, in fact the flaps of the inner wing, slowing down cause an effect of inertia which rotates the plane, however, being the inner wing braked by the flaps, it is necessary to raise slightly the flaps of the outer wing so as to have a slowdown in the rotation of the anode the axis of yaw, also controlled by the movement of the rudder. All of these actions cause the plane to tilt and turn. For simplicity we have decided to limit the rotation up to a roll angle of 90 degrees. The rotation is progressive and depends on the angle of roll, the more the inclination will be accentuated and the smaller the turning radius will be. On the right of the screen, there will be a roll angle indicator expressed in degrees, positive for counterclockwise rotations and negative for clockwise rotations. Rotation along the pitch axis is controlled by the W controls for the dive and the S controls for the cab. Also in this case the rotation is limited and less than 60 degrees. During pitch rotation, the outer flaps of the wings move in a coordinated and inverse manner to the rudder flaps. In fact, for the cabin you will need to raise the front of the aircraft and lower the thing. To do this, the outer flaps

[22]The roll angle is the angle that rotates around the longitudinal axis, that is, the axis that connects the tip with the tail.
[23]The pitch angle is the angle that rotates along the latitudinal axis, that is, the axis that connects the two wing ends.
[24]https://en.wikipedia.org/wiki/Adverse_yaw

of the wings will be lowered progressively, while the rudder flaps will be raised, thus causing a depression and lowering the tail. During the dive the opposite happens, i.e. the outer flaps of the wings are raised, while the rudder flaps are lowered. As for the roll, also the pitch has an indicator of the rotation, positive during the dive and negative during the cab. We decided not to give the possibility to mix the roll and pitch angles as the player would face configurations of difficult interpretation. We thought of a self-stable profile, which is for both roll and pitch, once released the command, the aircraft will return independently in the stable configuration. This greatly simplifies maneuverability and makes the game more accessible.

Listing 2: Manage commands

```
document.addEventListener("keydown", onDocumentKeyDown, false);
function onDocumentKeyDown(event) {

    if (!playFlag) return;

    switch (event.keyCode) {
        case 77: // m
            if (motors == 0) {
                motors = 1;
                speed_helic = 0.05;
                engine = setInterval(go_motors, 1);
                //sound on
                motor_sound.sound.play();
                setInterval(motion, 1);
                setInterval(manage_velocity, 40);
            }
            break;
        case 66: // b
            clearInterval(reset);
            flag = false;
            if (motors != 0 && motors != 5) {
                motors += 1;
                speed_helic += 0.05;
                motor_sound.sound.playbackRate += .7;
            }
            if (weels != 5) {
                weels += 1;
                speed_weels += 0.1;
            }
            break;
        case 78: // n
            clearInterval(reset);
            flag = false;
            if (motors != 0 && motors != 1) {
                motors -= 1;
                speed_helic -= 0.05;
                motor_sound.sound.playbackRate -= .7;
            }
            break;
        case 65: // a
```

```
    if (pitch < 0.05 && pitch > -0.05 && roll <= 90 && !ground && !emptyingTank &&
        !onLake) {
        clearInterval(reset);
        flag = false;
        model.rotateX(Math.PI / 300);
        roll += 180 / 300;
    }
    break;
case 68: // d
    if (pitch < 0.05 && pitch > -0.05 && roll >= -90 && !ground && !emptyingTank &&
        !onLake) {
        clearInterval(reset);
        flag = false;
        model.rotateX(-Math.PI / 300);
        roll -= 180 / 300;
    }
    break;
case 87: // w
    if (roll < 0.05 && roll > -0.05 && pitch <= 60 && !emptyingTank && !onLake &&
        !ground) {
        clearInterval(reset);
        flag = false;
        model.rotateZ(Math.PI / 400);
        pitch += 180 / 400;
    }
    break;
case 83: // s
    if (roll < 0.05 && roll > -0.05 && pitch >= -60 && !emptyingTank && (!ground ||
        vel > 150)) {
        clearInterval(reset);
        flag = false;
        model.rotateZ(-Math.PI / 400);
        pitch -= 180 / 400;
    }
    break;
case 67: // c
    if (carrello && !ground) {
        t = 0;
        s = 0;
        carrello = false;
        pressed_c = true;
        cart_sound.sound.play();
        setInterval(close_doors_ant, 10);
        setInterval(close_doors_back, 10);
    }
    break;
case 32: //space bar
    if (tank) {
        let timeout;
        pressed_bar = true;
```

```
            let r = roll.toFixed(0);
            let p = pitch.toFixed(0);
            if (r > -20 && r < 20 && p > -40 && p < 80 && (!onLake || height >
                height_difficulty)) {
                waterClock.start();
                tank = false;
                emptyingTank = true;
                water_sound.sound.play();
                moving_bar(0);
            }
            if(posizione_sopra_fuoco(model.position.x,model.position.z)){
                var distanzaDaFuoco = Math.sqrt(Math.pow((model.position.x -
                    firePosition[0]),2)
                    + Math.pow(model.position.z - firePosition[1],2));
                var quanto = 5 - Math.abs(distanzaDaFuoco)/100;
                if(quanto < 0)
                    quanto = 1;
                console.log(quanto);
                fire_extinguish(quanto);
                msg_id++;
                if (msg_id%3==0) msg_id=0;
            }
            clearTimeout(timeout);
            timeout = setTimeout(function() {
            pressed_bar = false;
            }, 5000);
        } else if (onLake && !emptyingTank) {
            moving_bar(1);
            tank = true;
        }
        break;
    }
};
```

Velocity

We have decided to equip the plane with five levels of speed. Initially it starts from a zero speed, with the engines off. Pressing the M button starts the engines, which causes the wheels to start (both front and rear) and the propellers to rotate (the rotation takes place in the opposite direction to prevent the aircraft from rotating due to the moment of inertia). Once the motors have been switched on, the speed can be controlled using keys B and N, which allow acceleration and deceleration respectively. Each time one of the two keys is pressed, the next (or previous) speed level is reached, up to a maximum level of 5 and a minimum level of 1. The speeds for each level are the following:

- **Level 1**: 171 Km/h
- **Level 2**: 206 Km/h
- **Level 3**: 263 Km/h

- **Level 4**: 320 Km/h
- **Level 5**: 376 Km/h

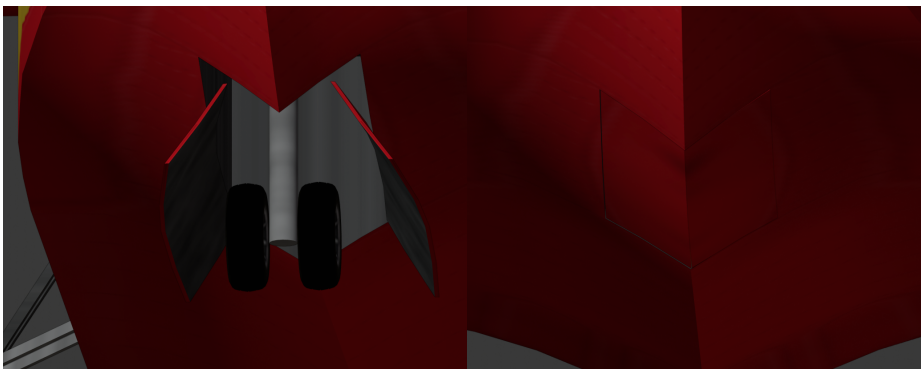
However, speeds are affected by the aircraft's trim, reducing in the case of a cab or increasing in the case of a dive. This was achieved by integrating the pitch angle into the manage_velocity function.

Listing 3: Function manage_velocity

```
function manage_velocity() {

    if (!playFlag) return;

    if (ground && motors > 0 && vel < 171) vel += 1.2;
    if (vel < 206+(Math.sin(pitch*Math.PI/180)*20) && motors > 1) vel += 0.67;
    else if (vel < 263+(Math.sin(pitch*Math.PI/180)*20) && motors > 2) vel += 0.5;
    else if (vel < 320+(Math.sin(pitch*Math.PI/180)*20) && motors > 3) vel += 0.5;
    else if (vel < 376+(Math.sin(pitch*Math.PI/180)*20) && motors > 4) vel += 0.33;

    else if (vel > 376+(Math.sin(pitch*Math.PI/180)*20)) vel -= 0.5;
    else if (vel > 320+(Math.sin(pitch*Math.PI/180)*20) && motors < 5) vel -= 0.5;
    else if (vel > 263+(Math.sin(pitch*Math.PI/180)*20) && motors < 4) vel -= 0.5;
    else if (vel > 206+(Math.sin(pitch*Math.PI/180)*20) && motors < 3) vel -= 0.5;
    else if (vel > 171+(Math.sin(pitch*Math.PI/180)*20) && motors < 2) vel -= 0.5;

    height = model.position.y*0.4;

}
```

Take-off

The take-off phase is one of the most delicate phases in the flight activity, the pilot must make the plane take as much speed as possible and then cabrate to take altitude. To make the take-off phase easier, we have prevented the player from making a dive or a turn during the initial take-off phase. This was possible thanks to the ground variable, a Boolean variable that is true when the aircraft is on the ground. A Bombardier-415 has a stall speed of about 150 km/h, so we made it impossible for the plane to take off at lower speeds. Being on the runway, in addition to the thrust of the propellers also has the acceleration due to the rotation of the wheels, so the acceleration during the ground phase of the takeoff is greater than that which occurs during the flight. Once the take-off is complete, it is essential that the pilot calls back the landing gear, especially as the landing gear would crash if it approached an aquatic surface. To close the landing gear, the player can press button C, which will activate two functions for closing the front and rear landing gear.



(a) Front gear open (b) Front gear close

Figure 3: Front gear during the closing of the landing gears

(a) Rear gear open (b) Rear gear close

Figure 4: Rear gear during the closing of the landing gears

Game Screens

Home Page

Figure 5: Home page with difficulty menu

The home page has 4 buttons in the middle of the screen:

- **PLAY GAME**: allows you to start the game.
- **INSTRUCTIONS**: provides a detailed explanation of the rules and commands.
- **COMMANDS**: illustrates the functions associated with the keys.
- **LIGHT MODE**: activates the light mode required for less powerful devices.

At the bottom you can adjust the difficulty, choosing through 4 levels that appear from a drop-down menu. The difficulties are as follows:

- **BEGINNER**: to be considered as a tutorial, the fire will not expand and you can fill the tank with a height less than or equal to 15 meters.
- **EASY**: For novice players, the fire will expand slowly and you can fill the tank with an altitude less than or equal to 10 meters.
- **NORMAL**: For players looking for a challenge, the fire will expand quickly and you can fill the tank with an altitude less than or equal to 6 meters.
- **HARD**: For experienced players who want an extreme challenge, the fire starts bigger and will expand very quickly, moreover you can fill the tank with an altitude less than or equal to 3 meters.

At the top left of the home screen there is a button with a loudspeaker that allows you to enable or disable the sound.

Loading Window

During the most intense uploads we have inserted a screen that gives an idea of the loading situation. The screen was downloaded from *Freefrontend*[25], we chose a simple screen that reflected the colors of the fire[26]. We wrote the following code to upload it:

Listing 4: Loading manager

```
const loadingManager = new THREE.LoadingManager( () => {

        const loadingScreen = document.getElementById( 'loading-screen' );
        loadingScreen.classList.add( 'fade-out' );
        loadingScreen.style.display = "none";
        loadingScreen.classList.remove( 'fade-out' );

        menu_music.muted = true;

        //show game window
        document.getElementById('game_id').style.display = 'block';

        if (volume) fire_sound.setVolume( 8 );
        else fire_sound.setVolume( 0 );

        playFlag = true;
        animate();
        setInterval(messages, 1000);
        startTime=clock.getElapsedTime();

        // optional: remove loader from DOM via event listener
        loadingScreen.addEventListener( 'transitionend', onTransitionEnd );
} );
```

[25]https://freefrontend.com/css-loaders/
[26]https://codepen.io/sashatran/pen/vRrxXw

Game Screen

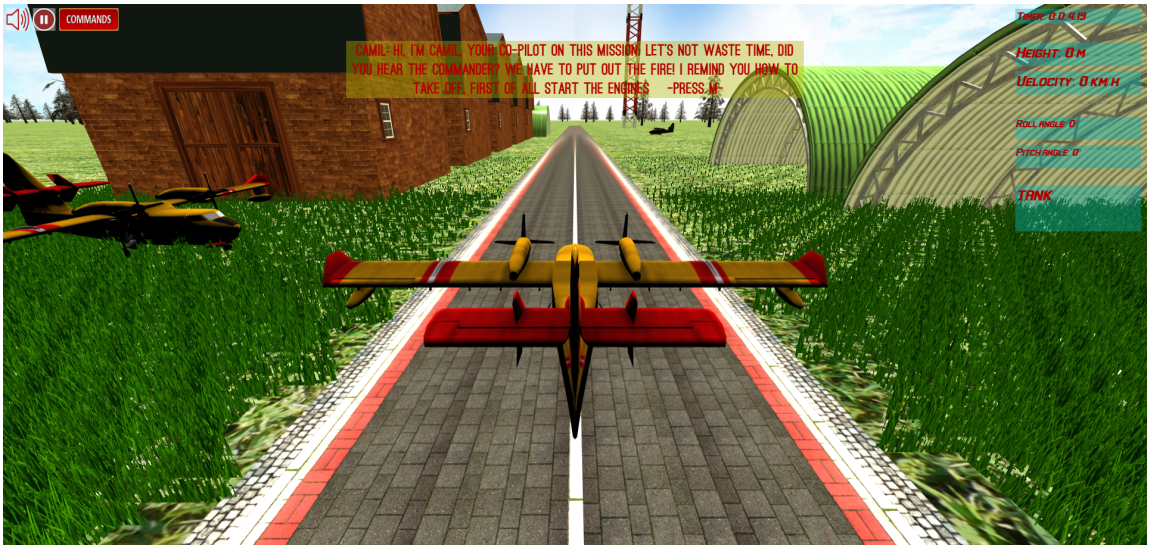


Figure 6: First scene of the game

At anytime the player can choose to mute or unmute the sounds, pause the game or check again the commands, using the buttons at the top-left of the screen. On the right he can monitor some important data, which change during the game: the elapsed time, the current height and speed, the orientation of the canadair and current tank conditions. Finally in the yellow box at the top he will receive usefull advice from the co-pilot, depending on the current situation and the actions taken.

Pause Screen

This screen allows the player to stop the game. In addition, he can decide to restart the game or come back to the home page (e.g. to change difficulty or switch to light mode).

In order to have a faster loading and a more performing game, we have decided to free the memory and reload the whole world only when you return to the initial menu and switch from light mode to full game or vice versa. In other cases, the canadair is reset to its initial position and a fire of a size consistent with the chosen level of difficulty is generated.

Victory Screen

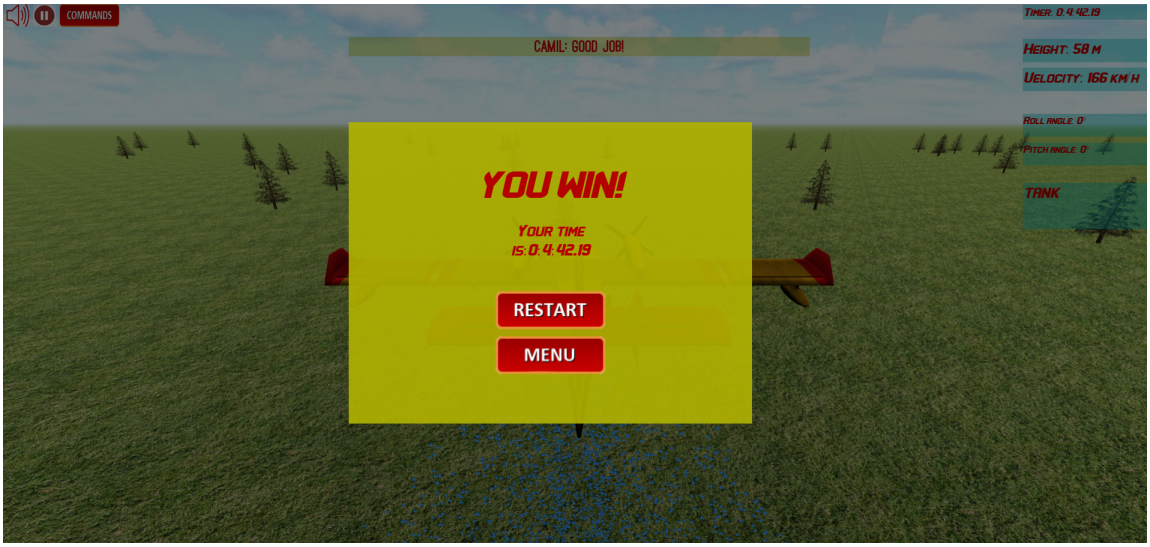


Figure 7: Victory screen

It appears once the fire is completely extinguished, and informs the player of the time taken. From here he can decide to restart the game or come back to the home page (e.g. to change difficulty or switch to light mode).

Game over Screen

It appears when something goes wrong, and tells the player the error made. From here he can decide to restart the game or come back to the home page (e.g. to change difficulty or switch to light mode).